The Pathfinding Benchmark (Extended Abstract)

Forest Agostinelli[1], Shahaf Shperberg[2], Ian Turner[1], Peng Fu[1], Rojina Panta[1], Misagh Soltani[1], Cale Workman[1], Ritvick Neerattil[1], Francisco León[1], Robert Vazquez[1], Amin Tavakoli[3], Pierre Baldi[4]

[1]University of South Carolina, USA
[2]Ben-Gurion University of the Negev, Israel
[3]California Institute of Technology, USA
[4]University of California, Irvine, USA

foresta@cse.sc.edu, shperbsh@bgu.ac.il, irturner@mailbox.sc.edu, pfu@cse.sc.edu, {rpanta, msoltani, workmant, neerattr}@email.sc.edu, jleon@mailbox.sc.edu, ryv@email.sc.edu, amint@caltech.edu, pfbaldi@ics.uci.edu

Abstract

Pathfinding problems are found throughout chemistry, mathematics, robotics, and computing. We present the Pathfinding Benchmark (PB), a work-in-progress benchmark of diverse and challenging pathfinding problems to evaluate the ability of state-of-the-art pathfinding algorithms and inspire new pathfinding algorithms. PB contains pathfinding problems from chemical reaction mechanism pathfinding, quantum circuit synthesis, theorem proving, multi-agent pathfinding, and combinatorial puzzles. Furthermore, PB pushes the boundaries of what should be posed as a pathfinding problem by posing language modeling, image generation, and training neural networks as pathfinding problems.

Preliminaries

A **pathfinding domain** is comprised of states, actions, a transition function, $s' = T(s, a)$, that maps states and actions to next states, and a transition cost function, $c(s, a)$, that maps states and actions to positive transition costs. This can be represented as a weighted directed graph, where nodes represent states, edges represent actions that transition between states, and edge weights represent transition costs (Pohl 1970). A **pathfinding problem instance** is defined by a domain, a start state, and a goal (defined as the set of states considered goal states). The objective is to find a sequence of actions that transforms the start state into a goal state, with a preference for lower-cost paths (i.e., the sum of transition costs).

The Pathfinding Benchmark (PB)

Domain Definition Language

Defining a pathfinding domain requires a transition function, a transition cost function, and a goal test function. Training a heuristic further requires sampling problem instances, actions for random exploration, and goals for RL methods like hindsight experience replay (Andrychowicz et al. 2017). Because PB defines domains in Python without assuming an underlying structure, it seamlessly integrates external libraries, such as Lean for math or RDKit for chemistry.

Domains

The domains found in PB challenge both traditional planners based on standardized representations and those based on machine learning. Furthermore, since goal tests are black-box functions, goal specifications can be so high-level that they need not specify any state variables. For example, in program synthesis, goals can be specified as a set of input/output examples, which is agnostic to the language used to implement the program. Some domains may require search in an action space of a very large or infinite size.

Combinatorial puzzles. Combinatorial puzzles, such as the Rubik's cube, Lights Out, the sliding tile puzzle, and Sokoban, provided motivation for learning heuristic functions with deep neural networks (DNNs) (Agostinelli et al. 2019). We build on this with puzzles of arbitrary size (e.g., a 7x7x7 Rubik's cube), as well as other puzzles such as the pancake puzzle and the number link puzzle.

Theorem proving. A mathematical theorem is specified by its current proof state. The goal is to reach a terminal state with no remaining logical subgoals, guaranteeing a complete formal proof. Actions apply formal logical steps or mathematical transformations—such as algebraic substitutions or invoking lemmas—to reduce the current state into simpler subtasks. Because the space of valid actions is practically infinite, candidate steps are generated using pre-trained language models. These candidate actions are then executed within a formal verification environment, such as Lean, to iteratively compute subsequent states.

Quantum circuit synthesis. A quantum algorithm can be specified as a $2^n \times 2^n$ matrix, where n is the number of qubits. We pose this as a goal that implicitly represents a set of quantum circuits that implement the algorithm. Actions add gates to the circuit, where some gates are more costly than others. While solving problems with only a couple of qubits has practical applications due to the ability to decompose larger circuits into smaller ones, the domain can be given any integer $n >= 1$ as an argument.

Chemical reaction mechanisms. A chemical reaction is composed of reaction mechanism steps that are not trivial to determine but are important to know for performing reactions in practice. The start state is the reactants, the goal is any set of molecules that contain the desired products, and

actions are reaction mechanism templates.

Multi-agent pathfinding. Multi-agent pathfinding is often posed as an explicitly multi-agent problem where conflicts are resolved by a centralized algorithm. Instead, we pose the problem as a single pathfinding problem with an action space of 5^N, where each of the N agents can go up, down, left, right, or stay in place.

Program synthesis. Implementing a concise program that produces a set of input/output pairs is a fundamental problem in computing. PB will pose program synthesis tasks, such as the abstraction and reasoning corpus (Chollet et al. 2024), as pathfinding problems, where states are programs, actions modify programs, and input/output pairs implicitly define a set of programs that produce the outputs given the inputs.

Large language model reasoning. The start state is an initial text prompt and actions append a single token from a massive vocabulary (e.g., $|\mathcal{A}| \approx 50,000$). The objective is to find a valid sequence of tokens (i.e., a reasoning chain) that produces a response satisfying a verifiable goal condition based on the prompt, such as yielding a correct mathematical answer or passing a trained goal-predictor network. Transitions incur a uniform cost to encourage concise reasoning. This domain challenges algorithms to replace standard autoregressive decoding with cost-minimizing search over massive discrete action spaces.

Symbolic regression. Finding equations using a given set of symbols that explain observed reality is crucial to understanding the world. States are symbolic equations; actions modify these equations; and goals are observed data that implicitly define a set of symbolic equations that fit the data.

Image generation. PB poses image generation as a pathfinding problem where states are images, actions modify pixels, and the goal test determines if the image comes from the target distribution. Since this goal test can be difficult to perform with DNNs as they struggle with out-of-distribution detection, we will create a goal test for handwritten digits using traditional computer vision techniques. Future work will leverage ensembles of DNNs to perform the goal test.

Machine learning. PB poses supervised and reinforcement learning as pathfinding problems where states are function architectures and parameters, and actions modify them. For supervised learning, goals are sets of input/output examples that implicitly define an accurate mapping function. For reinforcement learning, goals specify a minimum return that implicitly defines a policy function that obtains such a return. When posed as pathfinding problems, functions need not be differentiable and goal tests can be performed on a held-out validation set, forcing the heuristic function to consider the generalizability of the function.

Problem Instances

Problem instances will be created for each domain by drawing from instances of critical importance to that domain, such as often-used algorithms in quantum computing and landmark theorems that have been proven, as well as those that have yet to be proven. To allow for a more informative comparison of algorithms, simpler problems and problems generated by random walks will also be included.

Metrics

PB seeks to push the limits of state-of-the-art algorithms and is primarily evaluated on the following metrics, in order of importance: (1) solution success rate, (2) path cost, (3) inference time, (4) inference memory, (5) training time, and (6) training memory.

Baselines

The following baselines will be used to solve problem instances with a given time limit.

Uniform cost search. Uniform cost search needs only the definition of the problem instance, works with a black-box transition function and goal test, and requires no training.

Fast Downward. Fast Downward (Helmert 2006) requires the pathfinding problem to be defined in a standardized language where the transition function and goal test are white-box functions defined using formal logic. Heuristics can be quickly computed from this representation without training. However, not all domains can be easily represented using formal logic.

Deep reinforcement learning. For each domain, Deep RL learns a DNN heuristic function that generalizes across states and goals. Deep RL requires a representation of state-goal pairs that is compatible with the DNN being trained, but works with black-box transition functions and goal tests. The learned heuristic function is then used with heuristic search to solve problems.

Acknowledgments

The work of Shahaf Shperberg was supported by the Israel Science Foundation (ISF) grant #909/23, by Israel's Ministry of Innovation, Science and Technology (MOST) grant #1001706842, in collaboration with Israel National Road Safety Authority and Netivei Israel, awarded to Shahaf Shperberg, by BSF grant #2024614 awarded to Shahaf Shperberg, as part of a joint NSF-BSF grant with Forest Agostinelli. This material is based upon work supported by the National Science Foundation under Award No. 2426622.

References

Agostinelli, F.; McAleer, S.; Shmakov, A.; and Baldi, P. 2019. Solving the Rubik's cube with deep reinforcement learning and search. *Nat. Mach. Intell.*, 1(8): 356–363.

Andrychowicz, M.; Wolski, F.; Ray, A.; Schneider, J.; Fong, R.; Welinder, P.; McGrew, B.; Tobin, J.; Abbeel, O. P.; and Zaremba, W. 2017. Hindsight experience replay. In *Advances in Neural Information Processing Systems*, 5048–5058.

Chollet, F.; Knoop, M.; Kamradt, G.; and Landers, B. 2024. Arc prize 2024: Technical report. *arXiv preprint arXiv:2412.04604*.

Helmert, M. 2006. The fast downward planning system. *Journal of Artificial Intelligence Research*, 26: 191–246.

Pohl, I. 1970. Heuristic search viewed as path finding in a graph. *Artificial intelligence*, 1(3-4): 193–204.